Politecnico di Torino

Digital systems electronics
A.A. 2018/2019

Prof. G. Masera

Lab 03

2 Apr 2019

Berchialla Luca	236032
Laurasi Gjergji	238259
Mattei Andrea	233755
Lombardo Domenico Maria	233959

1 4-bit Sequential RCA

1.1 Implementation

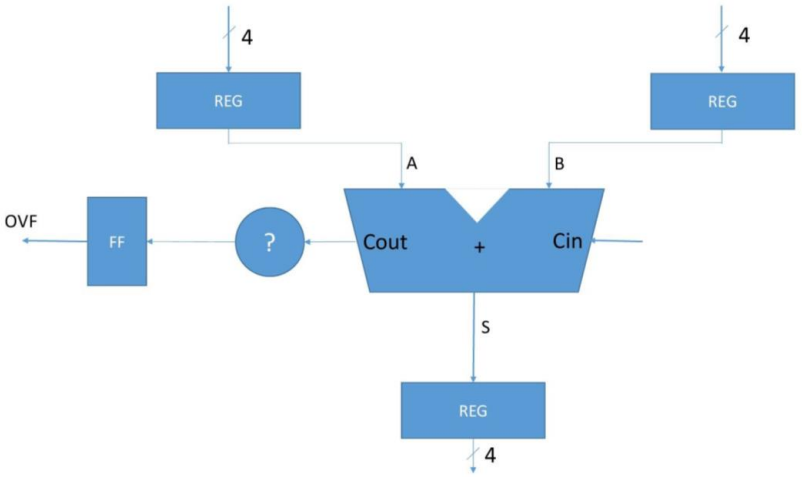

Figure 1: Top level entity

Here we had to implement a 4-bit Sequential Ripple Carry Adder. To build the circuit requested in $figure1$ several sub-circuits have been implemented.
As first point a Full Adder was implemented as shown in $figure\ 2a$. The 4-bit adder was built using four full adders in a ripple carry architecture. Its overflow signal is generated by a *xor* gate whose inputs are the $carry_{out}$ signals of the last two full adders.

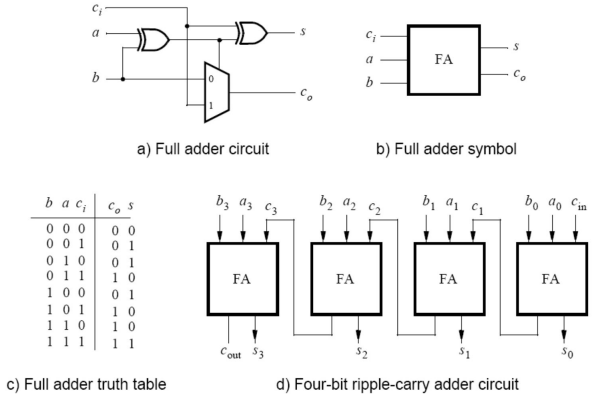

Figure 2: Full Adder

The Register and the Flip-Flop were implemented using the code provided in the

instructions.
A new 7-segments display decoder was needed in order to display properly 2's complement numbers. It uses two 7-segments displays to show respectively the sign and the magnitude of the number. The implementation was done by means of the $when - else$ statement following the truth table in $figure$ 3.

Display	Code	HEX(6)	HEX(5)	HEX(4)	HEX(3)	HEX(2)	HEX(1)	HEX(0)	Sign
0	0000	1	0	0	0	0	0	0	
1	0001	1	1	1	1	0	0	1	
2	0010	0	1	0	0	1	0	0	
3	0011	0	1	1	0	0	0	0	
4	0100	0	0	1	1	0	0	1	
5	0101	0	0	1	0	0	1	0	
6	0110	0	0	0	0	0	1	0	
7	0111	1	1	1	1	0	0	0	
-8	1000	0	0	0	0	0	0	0	-
-7	1001	1	1	1	1	0	0	0	-
-6	1010	0	0	0	0	0	1	0	-
-5	1011	0	0	1	0	0	1	0	-
-4	1100	0	0	1	1	0	0	1	-
-3	1101	0	1	1	0	0	0	0	-
-2	1110	0	1	0	0	1	0	0	-
-1	1111	1	1	1	1	0	0	1	-

Figure 3: 2's complement 7-segments decoder

The top level entity that implements the circuit is in the file $lab3\_es1.vhd$ where all the components needed are instantiated and connected together.
As required, the the inputs A and B have been assigned to SW_{3-0} and SW_{7-4} respectively, KEY_0 to the negated asynchronous reset input and KEY_1 to the clock. The magnitude and the sign of A are displayed in $HEX4$ and $HEX5$ respectively. The same is done for B and S in $HEX2, HEX3$ and $HEX0, HEX1$ respectively. The adder's overflow is shown by means of the red $LEDR_9$.

1.2 Testbench

To validate the correct behavior of the circuit a testbench was created. The *clock* signal was created using a process and its period has been fixed to $10ps$, the reset signal instead has a period of $115ps$, not a multiple of the clock, to verify that is asynchronous. It is active for $10ps$ and is generated just 10 times in order to have cleaner waveforms.
To test every possible combination of inputs it has been used a 8-bit counter which increments its value every $100ps$, the least significant 4 bits have been assigned to A and the others to B.
Since the the values of A, B and S coded for the 7-segments display are the output of the circuit in the testbench are present several if clauses that translate the output values to the 2's complement binary value of A, B and S.

1.3 Timing analysis

The maximum operating frequency of the circuit f_{max} has been determined with the help of Quartus Prime and TimeQuest and is $f_{max} = 650MHz$.
The longest path in the circuit in terms of delay is the one that starts from the MSB of a register where A or B are stored and arrives to the Flip Flop that stores the overflow signal. This path is longer than the other possible path that starts from one of the previous registers and arrives to the register where S is stored, because the overflow signal is calculated by a *xor* gate whose inputs are the $carry_{out}$ signals of the last two full adders. Therefore, with respect to the MSB of S, which goes directly to the register, the signal has to be processed by the *xor* gate before arriving to the flip flop.

2 4-bit Sequential Adder/Subtractor

2.1 Implementation

The circuit implemented in this section is a little modification of the circuit implemented in section 1. In particular the 4-bit full adder has been modified to make it perform also the subtraction. The $carry_{in}$ input is replaced by the $add_subtract$ input that enables the sum when its value is 0 and the subtraction when is 1. The B input is 2's complemented when $add_subtract = 1$ by assigning the B_i input of each full adder to the xor of B_i and $add_subtract$ and by assigning the $carry_{in}$ of the first full adder to $add_subtract$.

2.2 Testbench

The $add_subtract$ input was then assigned to SW_8 in the top level entity. In the testbench the counter has been modified now being 9-bit wide and its MSB is assigned to $add_subtract$.

2.3 Timing analysis

The maximum operating frequency of the circuit f_{max} has been determined as in section 1 and its value is $f_{max} = 600MHz$. f_{max} is lower for this circuit because the longest path now includes another level of logic that is between the output of the register where B is stored and the input of the full adder.

Therefore the longest path is the same as the previous section but it starts only from the previously mentioned register because only between it and the full adder there are the additional *xor* gates.

3 16-bit RCA, Carry-Bypass Adder and Carry-Select Adder

In this section will be presented three different implementations of 16 bit adders. Each of them will be simulated by means of a testbench, timing analysis will be performed as well. The different adders are tested over the same circuit architecture shown in the next *figure*. The 16-bit adder receives the 2 16-bit addends from 2 registers and sends the summing result to a third register. Finally the result and the addends are decoded into hexadecimal values.

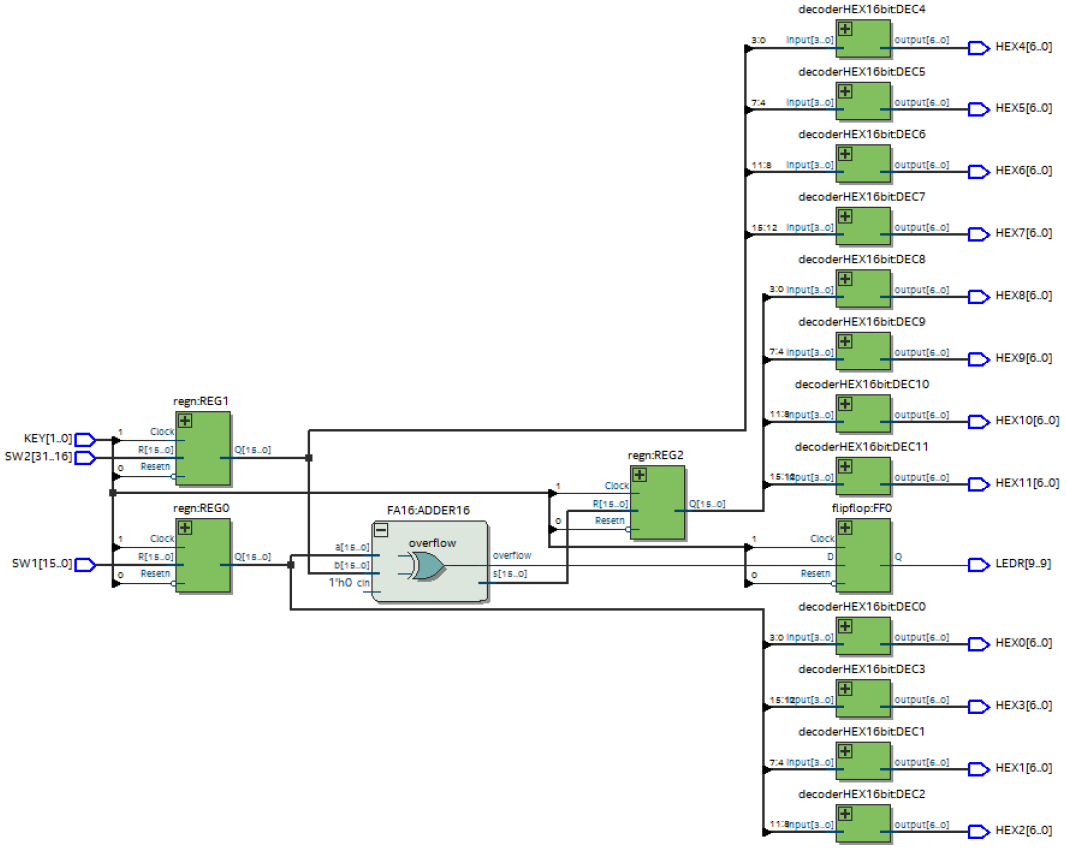

Figure 4: Architecture of the circuit including the general 16-bit adder

3.1 16-bit RCA

The first implemented architecture is the classical Ripple Carry Adder (RCA). As shown in the figure below the carry output of each FA becomes the carry input of the next FA.

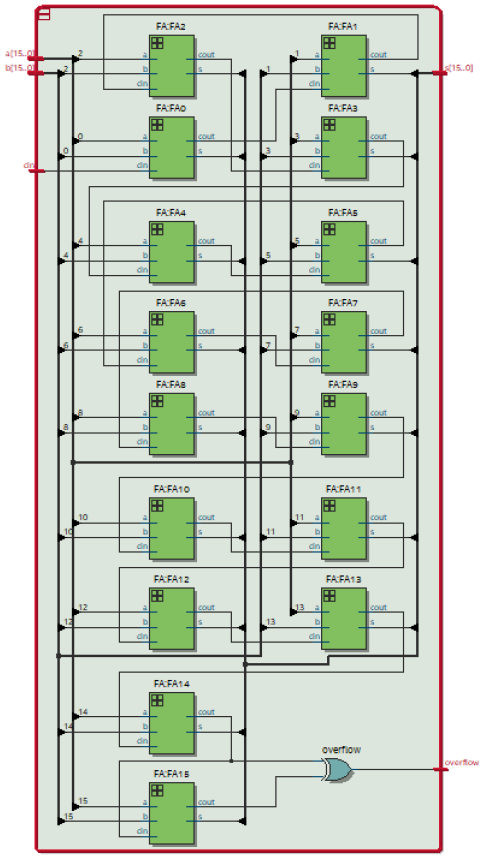

Figure 5: RCA adder architecture

The resulting thestbench is shown in the next image.

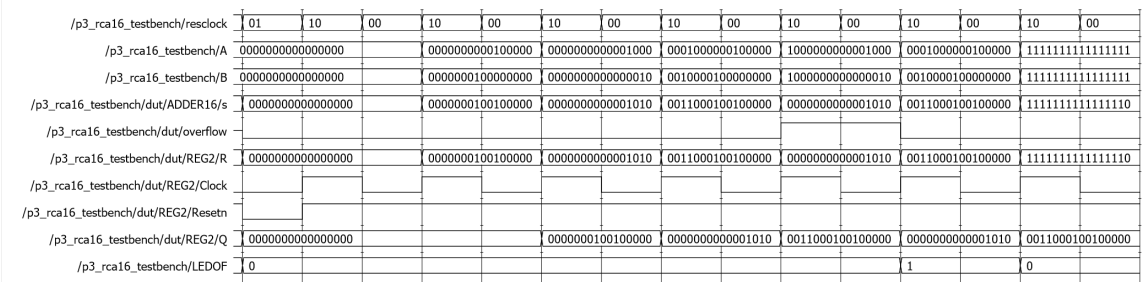

Figure 6: RCA testbench

The quartus timing analysis tool gives the results:

Fmax	Restricted Fmax	Clock Name
172.0 MHz	172.0 MHz	KEY[1]

	Property	Value
1	From Node	regn:REG0|Q[5]
2	To Node	flipflop:FF0|Q
3	Launch Clock	KEY[1]
4	Latch Clock	KEY[1]
5	Data Arrival Time	10.124
6	Data Required Time	5.310
7	Slack	-4.814 (VIOLATED)

meaning that the worst-case delay path is $5.814ns$ long, corresponding to a maximum usable frequency of $172MHz$.

3.2 16-bit Carry Bypass adder

The circuit shown below implements a 16-bit carry bypass adder. This architecture creates an alternative path for the carry of each block of full adders. If the output carry equals the input carry then it might be propagated resulting in a faster operation.

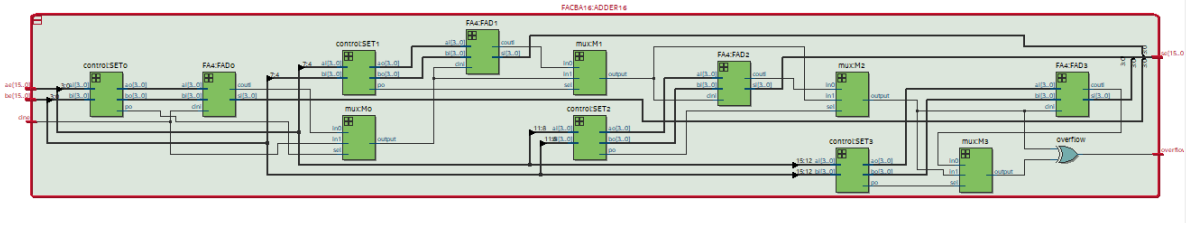

Figure 8: 16-bit Carry Bypass Adder

The control blackbox shown below checks the condition of carry propagation enabling a MUX that select the right carry path.

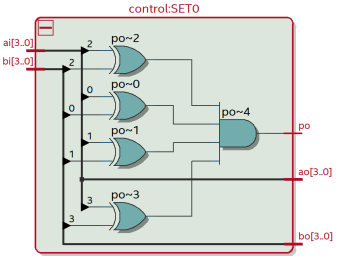

Figure 9: control blackbox

As before a testbench has been generated validating the design:

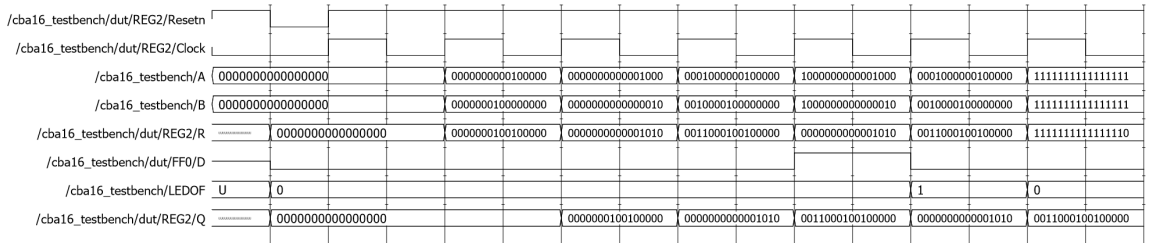

Figure 10: CBA testebench

This time quartus timing analysis tool returns the results:

Fmax	Restricted Fmax	Clock Name
218.91 MHz	218.91 MHz	KEY[1]

Property	Value
From Node	regn:REG0|Q[1]
To Node	flipflop:FF0|Q
Launch Clock	KEY[1]
Latch Clock	KEY[1]
Data Arrival Time	8.868
Data Required Time	5.300
Slack	-3.568 (VIOLATED)

meaning that the worst-case delay path is $4.568ns$ long, corresponding to a maximum usable frequency of $218.91MHz$, much higher than the first architecture.

3.3 16-bit Carry Select Adder

The circuit shown below implements a 16-bit carry select adder. This time the architecture doubles the number of FAs calculating the possibile results for each possible carry input condition. At the end the true result is generated using a set of MUX that select the 'already calculated' result corresponding to the given carry input.

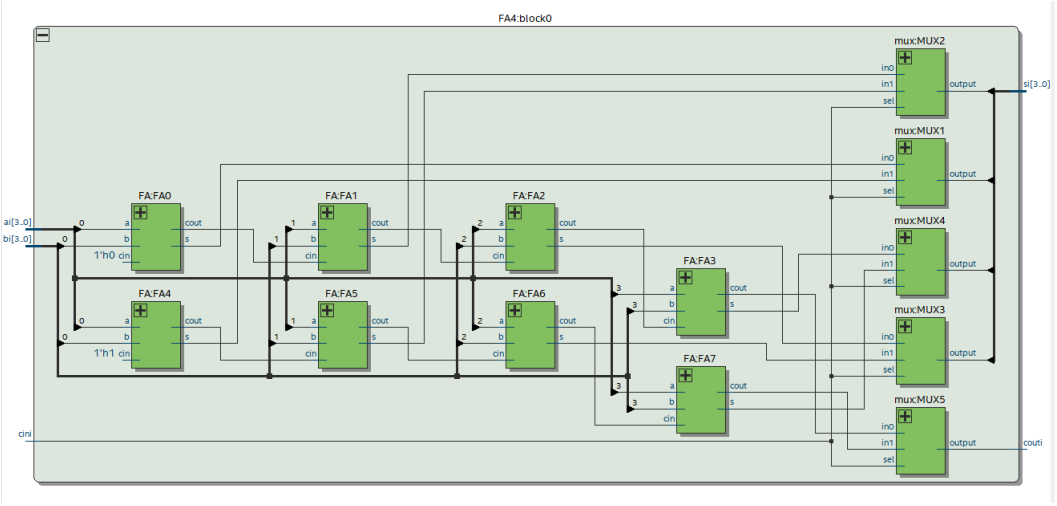

Figure 12: 16-bit Carry Select adder

As before a testbench has been generated validating the design:

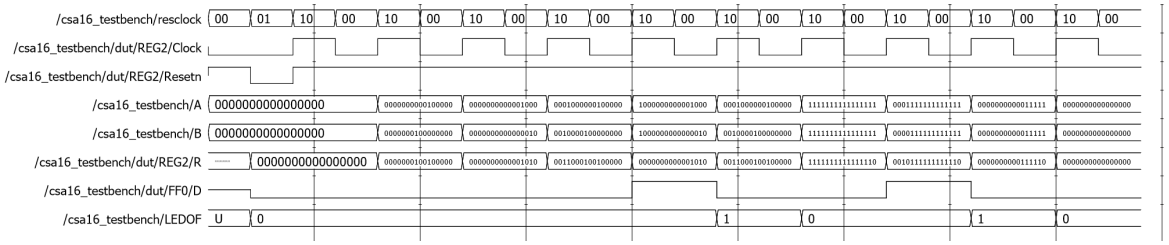

Figure 13: 16-bit CSA testebench

This time quartus timing analysis tool returns the results:

Fmax	Restricted Fmax	Clock Name
226.4 MHz	226.4 MHz	KEY[1]

1	From Node	regn:REG0|Q[0]
2	To Node	flipflop:FF0|Q
3	Launch Clock	KEY[1]
4	Latch Clock	KEY[1]
5	Data Arrival Time	8.692
6	Data Required Time	5.275
7	Slack	-3.417 (VIOLATED)

meaning that the worst-case delay path is $4.417ns$ long, corresponding to a maximum usable frequency of $226.4MHz$, this result is similar to the CBA adder but still the implemented 16-bit CSA returns the fastest results.

4 Multiplier

The task of this part of the experience is design a 4-bit multiplier using VHDL, the circuit implemented is based on the one in *Figure* 4. The input of the circuit are the

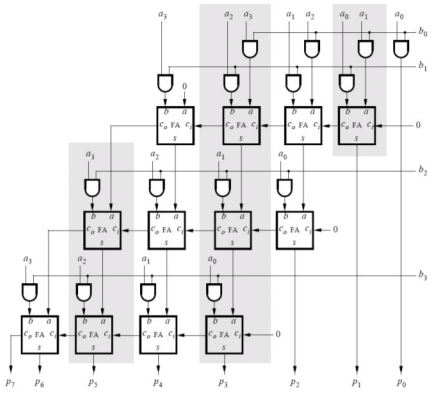

Figure 15: the circuit implemented for the multiplier

DE10 switch from 0 to 7, the first four represent the first number while the remaining compose the second number. The output of the circuit are the first 4 seven-segments displays of the board that are used for displaying hexadecimal numbers. The first 2 displays show the 2 operands while the other 2 display the result. The overall architecture (*Figure* 5) of the circuit is composed by the multiplier (multiplier.vhd) and 4 bcd to seven-segments converter for driving the displays. The converters used were

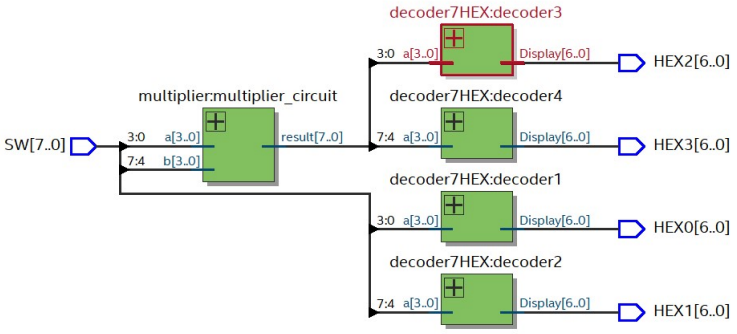

Figure 16: the RTL schematics of the overall circuit

the same of the previous parts, since their function is the same. For implementing the multiplier 2 type of components were used: 4-bit adders (adder.vhd) and and arrays (and_array.vhd). The 4-bit adders are composed of 4 full adders and they do the intermediate addition, like the 7-segments converters the full adder components were reused from previous parts. The and arrays perfoms the multiplication of one factor by one bit of the other.

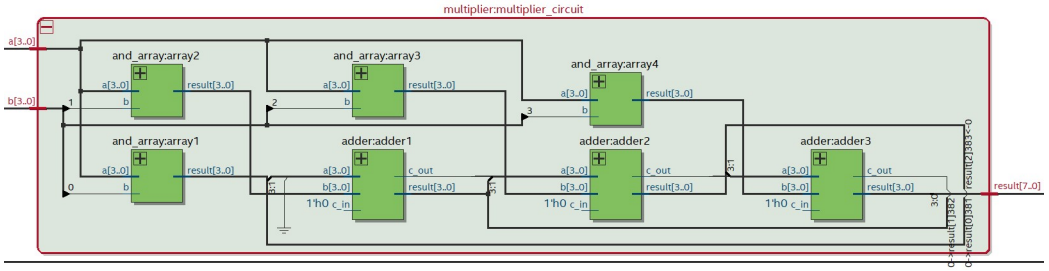

Figure 17: the RTL schematics of the multiplier component

4.1 Testing the circuit

The testing of single components was done manually using Modelsim since all of the components were either reused from previous points or trivial.
A testbench was used for testing the overall circuit, this testbench tests all the possible 256 pairs of inputs. Despite the big number of possible pairs reading the test result is practical as the operation of the circuit is just a simple multiplication.
Inputs were generated procedurally through 2 nested for loops and they were provided with a 8-bit vector that simulated the switches vector.
The tests of circuit done on the DE10 board confirmed the Modelsim simulations.